ELSEVIER

Theoretical Computer Science 177 (1997) 73–109

Theoretical Computer Science

Constructing process categories

J.R.B. Cockett[1], David A. Spooner[1,2,*]

University of Calgary, Department of Computer Science, 2500 University Drive, Calgary, Canada T2N 1N4

Abstract

A method of constructing process categories as generalized relations on a category of process models is presented. The construction may be viewed as a 2-functor, allowing structural properties of the process categories to be derived from the underlying structure of the model categories. In particular, this allows one to infer the presence of linear structure in a process category.

The construction yields Abramsky's category *SProc* when applied to any of the standard models of interleaved concurrency. *SProc* is also obtained as a process category upon the category of "sets in time" (i.e. trees) and this sheds new light on the analogy between *SProc* and "relations in time".

1. Introduction

Abramsky et al. [1] have proposed *interaction categories* as a new semantic paradigm. These categories take concurrent processes as morphisms between interface specifications and take composition to be process interaction at a shared interface. This treatment of processes as morphisms provides a type discipline for process construction. As in functional programming, this discipline may be used to facilitate correctness arguments for concurrent system implementations.

A key example of an interaction category is the category *SProc* of synchronous processes [1]: its objects are trace specifications and its morphisms are strong bisimularity classes of transition systems whose traces lie within their interface specifications. One aspect of the original formulation of *SProc* – and of interaction categories in general – is that morphisms are chosen to ensure that each has a canonical representative, thus avoiding the issue of process equivalence: in [1], Aczel's synchronization trees [2] are chosen to represent strong bisimulation classes of transition systems. In order to present and manipulate processes, however, it is often more desirable to use a state-based

* Corresponding author. E-mail: spooner@cpsc.ucalgary.ca.

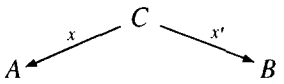
[1] Partially supported by NSERC, Canada.

[2] Partially supported by Lambda Software Corporation, Calgary, Canada.

PII S0304-3975(96)00235-6

formulation. Equality of processes then becomes an issue as such formulations inevitably allow many expressions of the "same" process.

Joyal et al. [14] have proposed a categorical view of bisimulation, which permits a uniform definition of bisimulation over a variety of models of concurrency. Roughly, objects A and B in a model category are *$\mathscr{X}$-bisimilar* when related by a span of morphisms



belonging to a *cover system*[3] $\mathscr{X}$, which is defined appropriately for each category. In the category of labeled transition systems, for example, the cover morphisms are those which reflect (as well as preserve) transitions: i.e. $f(s) \xrightarrow{b} t$ implies there exists a and s' such that $s \xrightarrow{a} s'$, $f(a) = b$, and $f(s') = t$. In the "fibre" over a chosen alphabet, two transition systems are strong bisimilar [18] exactly when related by a span of such cover morphisms.

This paper develops a construction of process categories as categories of generalized relations: one begins with a category of process models and a cover system expressing process equivalence; one then forms the category of spans quotiented by the cover system. The method is illustrated by constructing *SProc* upon a category of transition systems, thus giving an explicit treatment of process equality and justifying the use of state-based models.

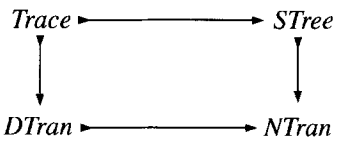
For an appropriate choice of functors and natural transformations, the process construction is 2-functorial. This is used to infer structure on process categories from related structure on the underlying (and inherently simpler) model categories. Regarding linear structure, the presence of finite limits, coproducts, and *multisets* in a model category may induce (respectively) compact-closure, biproducts, and storage in a process category. The latter two, however, depend critically on the properties of the cover system. In particular, we show that storage from multisets occurs in *SProc* for trace equivalence and not (as was suggested [4]) for bisimulation equivalence.

In constructing a process category, one has available a wide selection of model categories. Amongst the standard models of concurrency [21] there are several variations, which include:

- How the state space is modeled: the *system* models, such as transition systems, allow a process to reach the same state repeatedly through its evolution; the *behavior* models, such as trees, introduce maximal separation in the state space so that each state determines the process history at that point.
- Whether nondeterminism is modeled: the *branching time* models, such as synchronization trees and nondeterministic transition systems, model internal/nondeterministic choice; the *linear time* models, such as languages and deterministic transition systems, do not.

[3] Or system of *open* morphisms.

Sassone et al. [21] have established that the behavior models form coreflective subcategories of the corresponding system models, while the linear models form reflective subcategories of the corresponding branching models. In particular, the standard models of interleaved concurrency have the following relationship:



where the vertical arrows indicate coreflections, and the horizontal arrows indicate reflections. We show that all of these model categories give rise to the same process category – *SProc*. In fact, the only feature of the interleaving models which survives the process construction is "extension in time", even the explicit labeling of actions is unnecessary.

In the final section we briefly discuss a process category which is not equivalent to *SProc*. This process category is based on a model category for noninterleaved concurrency.

The paper is organized as follows: Section 2 discusses the construction of model categories, and develops the basic theory of cover systems. Section 3 presents the process construction and shows that it is 2-functorial. Furthermore, we consider the structure on a model category which may induce linear structure on a process category. Section 4 shows how relationships amongst various model categories induce equivalence of process categories. An abstract notion of behavior is developed and applied to the categories of transition systems. Section 5 outlines the construction of processes upon a model category for noninterleaving processes.

2. Model categories and cover systems

The construction of a process category requires a model category with pull-backs and a cover system: the models represent the intended dynamics of processes, and the cover system expresses the desired notion of equivalence.

As the structure of a process category generally arises from related structure in its model category, one is also interested in establishing the existence of structure such as limits and colimits in model categories.

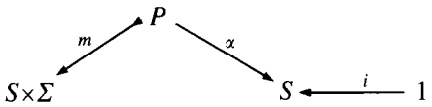
In this section we consider model categories which arise from sketches; we then discuss the theory of cover systems and various techniques for obtaining cover systems on model categories. For illustration we focus on a category of deterministic transition systems, which provides a simple basis upon which to construct *SProc*.

2.1. *Models for processes*

A model category for synchronous processes is naturally specified by a sketch. For example, consider the category of deterministic transition systems built upon a category

X with finite limits:

Definition 1. *Tran*(X) is the category of models in X of the following (finite limit) sketch:

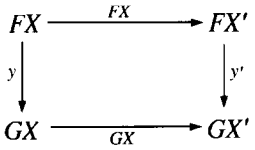


Here S is a state space with initial state i, Σ is an alphabet of actions, P is a subobject of $S \times \Sigma$ indicating the actions permitted at each state, and α determines the state change upon action.

While it is well known from sketch theory [22] that the category of models of a finite limit sketch has finite limits, we are concerned more specifically with limits/colimits which are given pointwise by those of the underlying category. To demonstrate conditions under which structure is inherited in this way by a model category, we consider the 2-categorical notion of inserter due to Kelly [15].

2.1.1. Inserters

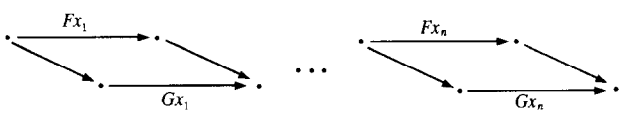
For F and G functors $X \to Y$, the *inserter* $F//G$ is the category whose objects are pairs $(X, y : FX \to GX)$ and whose morphisms $(X, y) \to (X', y')$ are those morphisms $x : X \to X'$ of X for which



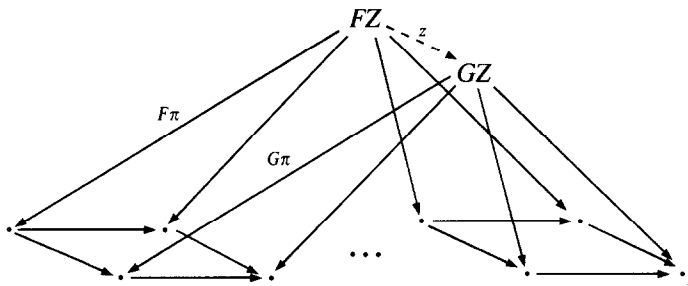
We say that $F//G$ has *pointwise limits* (resp. colimits) when the underlying functor $F//G \to X$ creates limits [17]. Let $\mathscr{D}$ be a graph.

Proposition 2. *$F//G$ has pointwise $\mathscr{D}$-limits whenever X has $\mathscr{D}$-limits and G preserves them.*

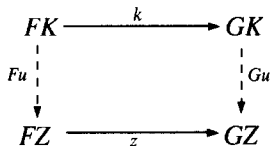
Proof. Any $\mathscr{D}$-diagram in $F//G$



determines a $\mathscr{D}$-diagram $x_1, \ldots, x_n$ in X. As G preserves $\mathscr{D}$-limits, forming the limit in X induces a cone in $F//G$



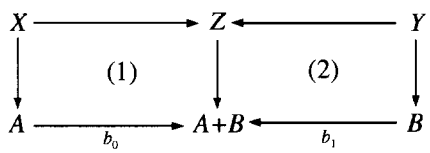
which sits uniquely above the corresponding cone in X. To see it is a limit cone, note that any other cone in $F//G$ determines also a cone in X. The mediating morphism to the limit cone in X gives the mediating morphism in $F//G$:



with commutivity forced since GZ is the limit in Y. □

Dually, $F//G$ has pointwise $\mathscr{D}$-colimits whenever X has $\mathscr{D}$-colimits and F preserves them.

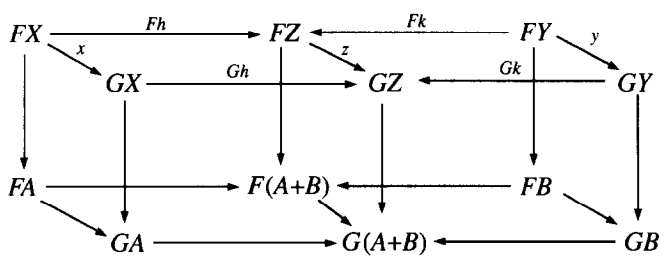
It is often desirable to have coproducts which are *extensive* [8]. A category with finite coproducts is extensive if given any commuting diagram



(1) and (2) are pullbacks if and only if the top row is a coproduct. An extensive category with finite limits is called *lextensive*.

Proposition 3. *$F//G$ is lextensive whenever X is lextensive, F preserves co-products and G preserves finite limits.*

Proof. The diagram of interest appears as follows in Y:



If (1) and (2) are pullbacks in $F//G$ then the extensive property in X makes (Z,h,k) a coproduct of X and Y in X, so (z,h,k) is a coproduct of x and y in $F//G$. Conversely, if the top row is a coproduct in $F//G$ then (Z,h,k) is a coproduct of X and Y in X and thus (1) and (2) are pullbacks in $F//G$. □

2.1.2. Limits in model categories

For $\mathscr{S}$ a sketch, we write $\underline{\mathscr{S}}$ for its underlying graph. Each node N in $\mathscr{S}$ determines a projection functor $F_N : X^{|Var(\mathscr{S})|} \to X$, where $Var(\mathscr{S})$ denotes the object variables in $\mathscr{S}$. If $\{a_1 : N_1 \to N_1', \ldots, a_m : N_m \to N_m'\}$ is the set of arrows in $\mathscr{S}$, then the model category $\underline{\mathscr{S}}(X)$ is the inserter

$$\langle F_{N_1}, \ldots, F_{N_m} \rangle // \langle F_{N_1'}, \ldots, F_{N_m'} \rangle.$$

As the pairing of any $\mathscr{D}$-limit (resp. colimit) preserving functors also preserves $\mathscr{D}$-limits (resp. colimits), securing pointwise limits (resp. colimits) in the model category $\underline{\mathscr{S}}$ amounts to placing constraints on the functors which are codomains (resp. domains) of arrows:

Corollary 4. *$\underline{\mathscr{S}}(\mathscr{X})$ has pointwise $\mathscr{D}$-limits whenever X has them and each codomain functor preserves them.*

Returning to the sketch *Tran* for transition systems we see that the functor $F_{S\times\Sigma} : X^3 \to X$ is given by $\langle \Pi_1, \Pi_2 \rangle ; ; \times$ which takes a triple (A_1, A_2, A_3) to the object $A_1 \times A_2$ in X, and the functor F_1 takes each triple to the final object. Thus $\underline{\mathscr{T}\nabla \dashv \backslash}(X)$ has finite limits, but does not necessarily have coproducts as F_1 does not preserve them (i.e. $1 \ncong 1+1$).

The question of when an arbitrary sketch has pointwise limits or colimits now becomes: when does the object introduced by forming a limit/colimit in $\underline{\mathscr{S}}(X)$ satisfy the constraints on models imposed by $\mathscr{S}$? Adding commutivity requirements is no problem: if two arrows are equal in each of the component models of a diagram in $\underline{\mathscr{S}}(C)$ then the corresponding arrows in the limit model also commute as each is the mediating morphism to a limit in X. Thus, for $\mathscr{S}$ a sketch without cones or cocones:

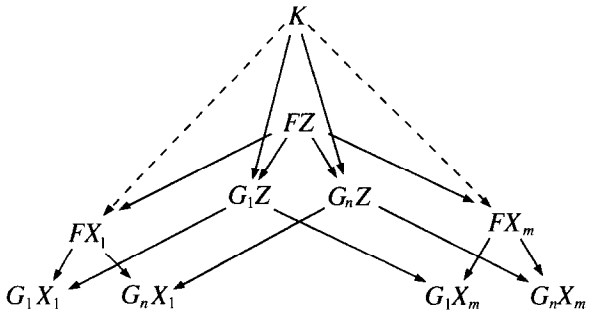
Proposition 5. *$\mathscr{S}(X)$ has pointwise $\mathscr{D}$-limits whenever $\underline{\mathscr{S}}(X)$ does.*

Maintaining limits in the presence of cones (resp. colimits in the presence of cocones) requires the additional condition that (the functor for) the object specified at the apex of each limit cone preserves the kind of limit being formed. For $\mathscr{S}$ a finite limit sketch.

Proposition 6. *$\mathscr{S}(X)$ has pointwise $\mathscr{D}$-limits whenever $\underline{\mathscr{S}}(X)$ has pointwise $\mathscr{D}$-limits and the domain functor of each of the cones specified in $\mathscr{S}$ preserves them.*

Proof. To simplify the exposition, we will regard $\underline{\mathscr{S}}$ as consisting of just the specified cone. If Z is a $\mathscr{D}$-limit in $\underline{\mathscr{S}}(X)$ then any cone over the model Z in X induces a cone over each of the constituent models and thus (as each is a limit cone) mediating

morphisms to each model:



This gives a $\mathscr{D}$-cone in X and, since the domain of the specified cone (viz. F) preserves $\mathscr{D}$-limits, gives a mediating morphism $K \to FZ$ which makes Z a limit cone in X. □

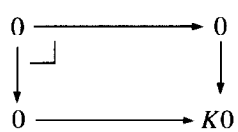
Considering again the sketch *Tran* of transition systems, we note that requiring the permission set m to be monic amounts to specifying that the square $1;m = 1;m$ is a pullback cone in X. With the machinery developed above we are now assured that the model category $Tran(X)$ has finite limits.

We now turn to the question of maintaining coproducts in the presence of limit cones. Note that a coproduct in $\underline{\mathscr{L}}(X)$ will not satisfy a final or product cone specification (viz. $1 \ncong 1 + 1$ and $(A + A') \times (B + B') \ncong A \times B + A' \times B'$).

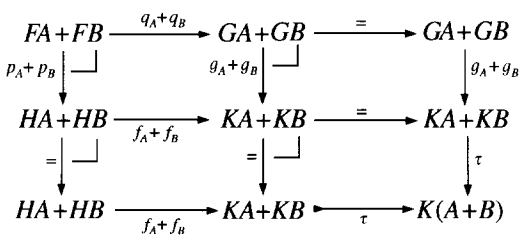
For X lextensive and $\mathscr{S}$ a sketch containing only pullback cones.

Proposition 7. *$\mathscr{S}(X)$ has pointwise coproducts provided $\underline{\mathscr{L}}(X)$ has pointwise coproducts and the codomain functor K of each specified pullback square has the canonical morphism $\tau : KA + KB \to K(A + B)$ monic in X.*

Proof. As the initial object is strict in X the following is a pullback



and thus any square in the initial model is a pullback. If $p;f = q;g$ is required to be a pullback in each model, the corresponding square in $A + B$ appears as follows



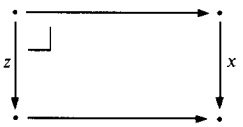
where we note that the upper left corner is a pullback as coproducts in X are stable. □

Although this does not help us obtain coproducts in the category of transition systems, we will see later that coproducts can be obtained in a related category of transition systems which have a separated initial state (see Definition 24).

2.2. Cover systems

Cover systems play a central role in the construction of process categories. Here we study some basic properties of cover systems, and provide various techniques for constructing cover systems – in particular, upon the model categories which arise from sketches.

In a category X with pullbacks, a collection $\mathscr{X}$ of morphisms of X is called a *cover system* when it contains all isomorphisms, is closed to composition, and is closed to pullback along arbitrary morphisms (viz. in the following pullback square, $x \in \mathscr{X}$ implies $z \in \mathscr{X}$).



These axioms state that a cover system $\mathscr{X}$ determines an equivalence relation on the objects of X: i.e. A and B are $\mathscr{X}$-equivalent when related by a span of morphisms in $\mathscr{X}$. In a category of models for concurrency, this may be regarded as a generalized bisimulation [14].

Examples of cover systems in any category X are the class X_1 of all morphisms, the isomorphisms $\mathscr{I}$, the retractions $\mathscr{R}$, and the monics $\mathscr{M}$. In a regular category, the regular epimorphisms form a cover system.

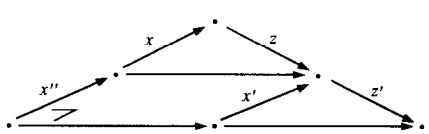
We say that a cover system $\mathscr{X}$ is *left-factor closed* if the fact that $f;g$ and g are in $\mathscr{X}$ implies that f is in $\mathscr{X}$. Similarly, $\mathscr{X}$ is *right-factor closed* if the fact that $f;g$ and f are in $\mathscr{X}$ implies g is in $\mathscr{X}$. For example: X_1 and $\mathscr{I}$ are both left and right-factor closed; $\mathscr{R}$ and the regular epimorphisms are right-factor closed; and $\mathscr{M}$ is left-factor closed.

Any cover system $\mathscr{X}$ can be closed by adding all its right factors, thus obtaining a cover system which is right-factor closed:

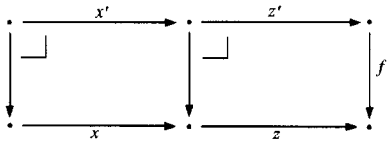
$$\mathscr{X}\downarrow \stackrel{\text{def}}{=} \{z \in X_1 \mid \exists x \in \mathscr{X}.x;\ z \in \mathscr{X}\}$$

Lemma 8. *$\mathscr{X}\downarrow$ is a right-factor closed cover system containing $\mathscr{X}$, which inherits left-factor closure from $\mathscr{X}$.*

Proof. We will say that x is a witness that $z \in \mathscr{X}\downarrow$ when both x and $x;\ z$ are in $\mathscr{X}$. Note that $\mathscr{X}\downarrow$ contains $\mathscr{X}$ (and thus all isomorphisms) as witnessed by the identity morphisms. To see compositionality, suppose x and x' are witness that z and z' belong to $\mathscr{X}\downarrow$:

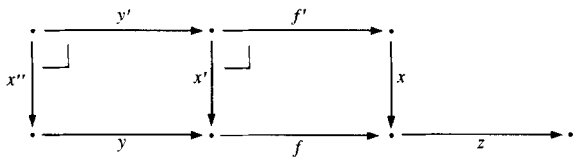


then $x'';x$ is witness that $z;z'$ is in $\mathscr{X}\downarrow$. To see stability, suppose x is witness that z is in $\mathscr{X}\downarrow$. Forming the pullback along an arbitrary f



yields the required witness x' for z'.

To see $\mathscr{X}\downarrow$ inherits left-factor closure, let z and $f;z$ belong to $\mathscr{X}\downarrow$ as witnessed by x and y, respectively, and consider the following diagram:

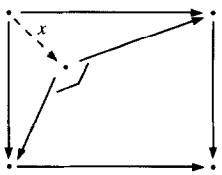


As $x'';y;f;z$ is in $\mathscr{X}$, we have $y';f'$ in $\mathscr{X}$ by left-factor closure and consequently f' is in $\mathscr{X}\downarrow$. Then f is in $\mathscr{X}\downarrow$ by right-factor closure. □

Of course if $\mathscr{X}$ is already right-factor closed, then $\mathscr{X}\downarrow$ is $\mathscr{X}$. We will see in Section 3.1 that both $\mathscr{X}$ and its right-factor closure determine the same equivalence on processes.

An obvious means of generating new cover systems from existing ones is by taking their intersection: if $\mathscr{X}_1$ and $\mathscr{X}_2$ are cover systems, then $\mathscr{X}_1 \cap \mathscr{X}_2$ is a cover system which is left- (resp. right) factor closed whenever either $\mathscr{X}_1$ and $\mathscr{X}_2$ are left- (resp. right) factor closed. Although the union of cover systems is not necessarily a cover system, it becomes so by simply closing under composition (although left/right-factor closure is not preserved).

Two further techniques for obtaining cover systems use the following generalization of pullback square. A commuting square in X is an *$\mathscr{X}$-pullback* if the induced morphism to the inscribed pullback is in $\mathscr{X}$:



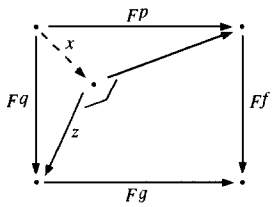
Clearly, f belongs to $\mathscr{X}$ if and only if the square $f;1 = f;1$ is an $\mathscr{X}$-pullback.

Any functor which takes pullbacks to cover-pullbacks induces a cover system on the domain of the functor.

Lemma 9. *If $F: Y \to X$ takes pullbacks to $\mathscr{X}$-pullbacks then $F^{-1}(\mathscr{X})$ is a cover system on Y which inherits left-factor closure from $\mathscr{X}$.*

Proof. As functors preserve isomorphisms and composition, the proposed cover system contains all isomorphisms and is closed to composition. To see stability, suppose Ff

is in $\mathscr{X}$ and that $p; f = q; g$ is a pullback in Y. Then in the diagram

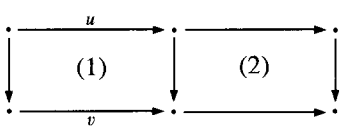


both x and z are in $\mathscr{X}$ and thus Fq is in $\mathscr{X}$ as required. The fact that F preserves composition forces $F^{-1}(\mathscr{X})$ to inherit left-factor closure from $\mathscr{X}$. □

As an example, note that $Tran(X)$ is fibred over X and that the fibration functor (which takes a transition system to its alphabet) is stable. One then obtains a cover system in $Tran(X)$ consisting of all morphisms whose label component is an isomorphism. A similar situation is found in [11] where the cover system for weak bisimulation on the asynchronous process category, $ASProc$, is obtained from the cover system for strong bisimulation on $SProc$.

To describe the technique for obtaining cover systems on model categories which arise from sketches, we will require a lemma concerning the behavior of $\mathscr{X}$-pullbacks. This result will also be valuable in many subsequent proofs.

Lemma 10. *Consider the following commuting diagram in X*:



(i) *the outer square is an $\mathscr{X}$-pullback whenever* (1) *and* (2) *are $\mathscr{X}$-pullbacks*;

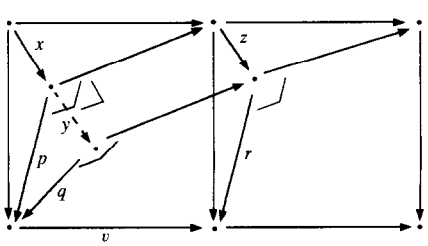
(ii) (1) *is an $\mathscr{X}$-pullback whenever* (2) *is a pullback and the outer square is an $\mathscr{X}$-pullback*;

(iii) *if $\mathscr{X}$ is left-factor closed then* (1) *is an $\mathscr{X}$ pullback whenever the outer square and* (2) *are $\mathscr{X}$-pullbacks*;

(iv) *if* (1) *is an $\mathscr{X}$-pullback whenever the outer square and* (2) *are $\mathscr{X}$-pullbacks, then $\mathscr{X}$ is left-factor closed*;

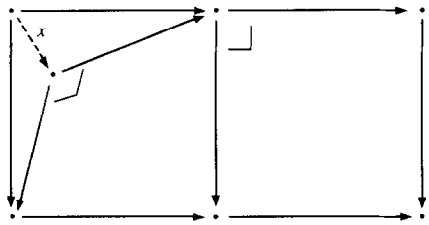
(v) *if the outer square is an $\mathscr{X}\downarrow$-pullback and u and v belong to $\mathscr{X}\downarrow$ then* (2) *is an $\mathscr{X}\downarrow$-pullback.*

Proof. (i) Let x and z witness that (1) and (2) are $\mathscr{X}$-pullbacks.

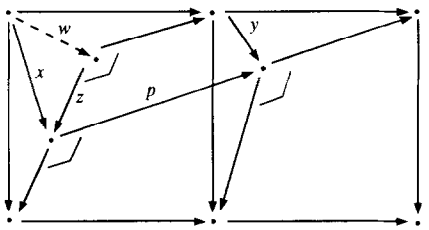


Forming the pullback of r and v, one obtains the pullback inscribed in the entire square; the induced morphisms to this pullback is x; y. As y is a pullback of z it belongs to $\mathscr{X}$ and thus x; y belongs to $\mathscr{X}$ also.

(ii) The induced morphism to the pullback inscribed in (1) is also the induced morphism to the pullback inscribed in the outer square.

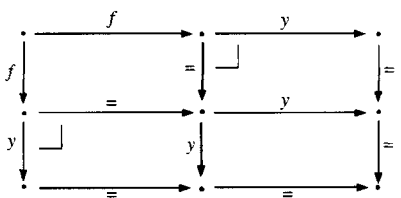


(iii) Suppose $\mathscr{X}$ is left-factor closed and let x and y witness that the outer square and (2) are $\mathscr{X}$-pullbacks.



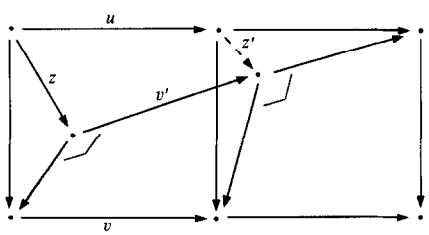
Forming the pullback of y along p gives a pullback inscribed in (1). The induced morphism w then belongs to $\mathscr{X}$ since x and z belong to $\mathscr{X}$.

(iv)Suppose $f; y$ and y are in $\mathscr{X}$. Using parts (i) and (ii) above, the top left square of



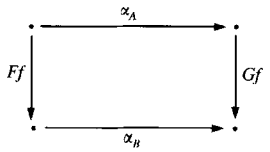
is a $\mathscr{X}$-pullback and thus f is in $\mathscr{X}$.

(v) In the following diagram



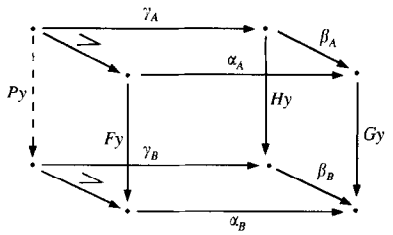
z' is a right-factor $z; v'$ and thus belongs to $\mathscr{X}\downarrow$. □

Let F and G be functors $Y \to X$, and f any morphism of X. We say that a natural transformation $F \Rightarrow G$ is *$\mathscr{X}$-cartesian for f* if the naturality square associated with f is an $\mathscr{X}$-pullback:



Proposition 11. *The natural transformations $F \Rightarrow G$ which are $\mathscr{X}$-cartesian for f form a cover system on Func(Y,X); this cover system is left-factor closed iff $\mathscr{X}$ is left-factor closed.*

Proof. The proposed cover system contains natural isomorphisms as these are cartesian, and is closed to composition by Lemma 10(i). To see that it is closed to pullback, suppose α is $\mathscr{X}$-cartesian for y and that γ results from pulling back α along arbitrary β :



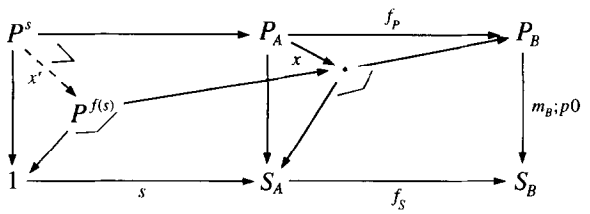
As the front face above is an $\mathscr{X}$-pullback, the rear face is also by Lemma 10(i) and (ii). Thus γ is $\mathscr{X}$-cartesian for y.

The result concerning left-factor closure is given directly by Lemma 10(iii) and (iv). □

As the category of models of a sketch $\mathscr{S}$ is essentially such a functor category, we can obtain a cover system by choosing any arrow σ of $\mathscr{S}$:

Corollary 12. *The class of morphisms of $\mathscr{S}(X)$ which are $\mathscr{X}$-cartesian for σ form a cover system; this cover system is left-factor closed iff $\mathscr{X}$ is left-factor closed.*

For example, in the category $Tran(X)$ of transition systems we define $\exists^{\mathscr{X}}$ to be the class of morphisms which are $\mathscr{X}$-cartesian for $m; p_0$. These morphisms are *locally $\mathscr{X}$* in the following sense:



If $f : A \to B \in \exists^{\mathscr{X}}$ then at any states s of A, the transitions from s (viz. P^s) map via a morphism of $\mathscr{X}$ to the transitions from $f(s)$. For example, the morphisms of $\exists^{\mathscr{R}}$ are local retractions: at each state s of A, every transition from state $f(s)$ in B has at least one corresponding transition from s in A. It is the cover system $\exists^{\mathscr{R}}$ which will give bisimulation equivalence upon the category *SProc* of synchronous processes. The local isomorphisms $\exists^{\mathscr{I}}$ will also be important later when we consider equivalence of various process categories.

3. Process categories

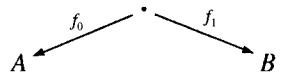
This section presents the construction of process categories as categories of generalized relations: given a model category with pullbacks and a cover system, one forms the bicategory of spans and then quotients by the cover system. We show that the construction may be seen as 2-functorial. This means the functorial structure in the model category (provided it interacts well with the cover system) is transmitted to the process category. In particular, we show how a process category may acquire the structure of a linear category.

For illustration we consider Abramsky's category *SProc*, which arises from the deterministic transition systems and cover system for bisimulation of the previous section.

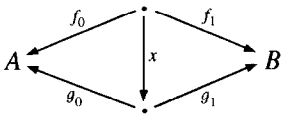
3.1. The process construction

Given a category X with pullbacks, the bicategory $Span(X)$ of spans in X [6] is given as follows:

- the 0-cells A are those of X;
- the hom-category $Span(X)(A,B)$ has as objects $f : A \rightsquigarrow B$ spans



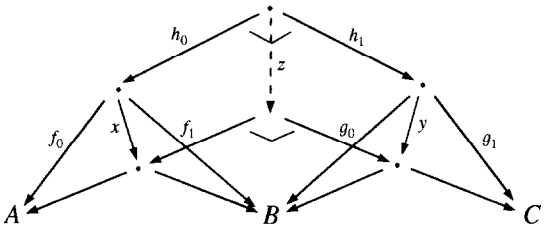
in X, and as arrows $f \Rightarrow g$ the morphisms x of X for which



commutes in X;

- horizontal identities $i_A : A \rightsquigarrow A$ are given by spans $(1_A, 1_A)$ of identities;
- horizontal composition is given by pullback – i.e. the composite $f;;g$ of 1-cells $f : A \rightsquigarrow B$ and $g : B \rightsquigarrow C$ below is the span $(h_0; f_0, h_1; g_1)$, and the composite $x;;y$ of 2-cells $x : f \Rightarrow f'$ and $y : g \Rightarrow g'$ is the induced 2-cell $z : f;;g \Rightarrow f';;g'$ below.[4]

[4] We will generally use the notation $A \rightsquigarrow B$ for 1-cells only to avoid ambiguity when simultaneously discussing arrows of the model category.



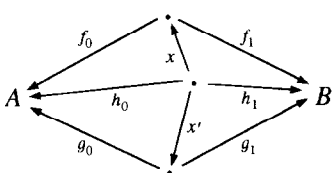
To develop some intuition for processes as spans, consider again the model category $Tran(X)$ of transition systems. The endpoints of a span are its interface specifications, which indicate the allowable actions at each state of the interface. The apex of a span is the (hidden) implementation of a process, and the span legs determine the visible effect of each process transition. Forming the pullback of spans yields a composite process whose states are the pairs of component states and whose transitions are those pairs of component transitions which synchronize at the shared interface. Finally, although transition systems here are deterministic, the emergent processes are not since a transition in the interface may be implemented by several alternative transitions in the apex.

We now consider how to quotient a process bicategory by an equivalence. Given any cover system $\mathscr{X}$ on X, one can restrict the 2-cells of $Span(X)$ to lie within $\mathscr{X}$ and obtain a sub-bicategory which we will call $Span(X)_{\mathscr{X}}$.

Proposition 13. *$Span(X)_{\mathscr{X}}$ is a bicategory.*

Proof. The cover system axioms ensure that the restricted hom-categories remain categories, and further that $Span(X)_{\mathscr{X}}$ is closed to horizontal composition and retains the isomorphisms winessing the unit and associativity axioms of horizontal composition. □

From any bicategory, one obtains a category by quotienting the 1-cells by 2-cell connections (see [19]). When the hom-categories have pullbacks (as $Span(X)_{\mathscr{X}}(A,B)$ does), this amounts to equating 1-cells whenever they are related by a span of 2-cells:



It is in this way we obtain the *process category* $Proc(X,\mathscr{X})$.

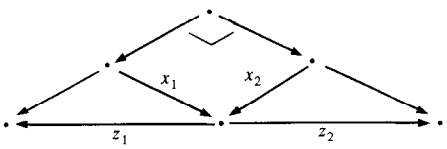
Considering transition systems and the cover system $\exists^{\mathscr{R}}$ for bisimulation, we see that two processes are equated in $Proc(X,\exists^{\mathscr{R}})$ when there is an (unlabeled) bisimulation on the apexes for which related transitions are identified in the interface. In Section 4.2

we will show that Abramsky's category *SProc* [1] is equivalent to the process category constructed upon *Tran*(*Set*) with respect to bisimulation. Thus for an arbitrary category X with finite limits, we will refer to $Proc(Tran(X), \exists^{\mathscr{R}})$ as $SProc(X)$.

More familiar examples of the process construction are the categories of spans and relations: $Span(X)$ is $Proc(X, \mathscr{I})$ and, for E a regular category with regular epimorphisms $\mathscr{E}$, $Rel(E)$ is $Proc(E, \mathscr{E})$. Note that in both of these examples, the cover system is right-factor closed. However:

Proposition 14. *$Proc(X, \mathscr{X})$ is equivalent to $Proc(X, \mathscr{X}\downarrow)$.*

Proof. Any $\mathscr{X}$-bisimulation is an $\mathscr{X}\downarrow$-bisimulation as $\mathscr{X} \subseteq \mathscr{X}\downarrow$. Conversely, one obtains an $\mathscr{X}$-bisimulation from an $\mathscr{X}\downarrow$-bisimulation



by pulling back the witnesses. □

Note that $\exists^{\mathscr{R}}$ is not right-factor closed due to the presence of unreachable states. However, when we restrict attention to the reachable and maximally separated transition systems (Section 4.2) we will see that bisimulations are right-factor closed.

3.1.1. Process isomorphisms

One may reasonably ask: what are the isomorphisms in a process category? First note that a process (f_0, f_1) is an identity in $Proc(X, \mathscr{X})$ if and only if there exist x and y in $\mathscr{X}$ such that $y; f_0 = x = y; f_1$. For example, any process (x, x) with x in $\mathscr{X}\downarrow$ is an identity.

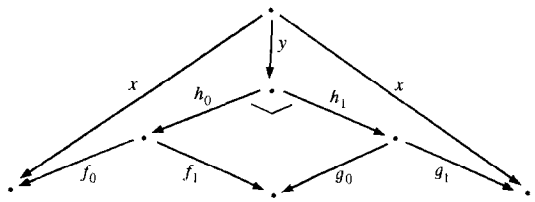
It is tempting to think that the process isomorphisms are just the spans with both legs taken from $\mathscr{X}\downarrow$. However, spans of covers need not be isomorphisms: for A and B nonempty sets, the chaotic relation $A \leftarrow A \times B \rightarrow B$ is such a span and is certainly not an isomorphism in *Rel*.

Unfortunately a process isomorphism need not have its legs covers unless – like the retractions or regular epimorphisms – $\mathscr{X}$ is closed to division on the right (i.e. $f; g \in \mathscr{X}$ implies $g \in \mathscr{X}$). If, however, a span of $\mathscr{X}$ maps is a process isomorphism then its inverse must be its span reversal:

Lemma 15. *If (f_0, f_1) is iso in $Proc(X, \mathscr{X})$ with $f_i \in \mathscr{X}\downarrow$, then $f^{-1} = f^{\circ}$.*

Proof. Let g be the inverse of $f : A \rightsquigarrow B$. As $f_1 \in \mathscr{X}$ and g is a right inverse, we have

$g_1 \in \mathscr{X}\downarrow$:

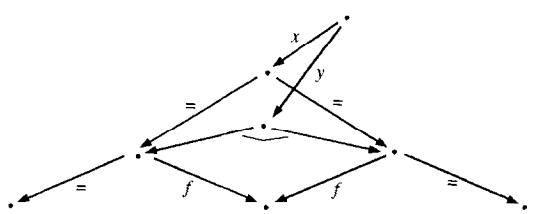


Similarly $g_0 \in \mathscr{X}\downarrow$. Thus $(y; h_0, y; h_1)$ is a $\mathscr{X}\downarrow$-bisimulation of f° and g. □

Although we cannot offer a simple characterization of isomorphisms in an arbitrary process category, we can characterize those cover morphisms which become isomorphisms:

Lemma 16. *For $f \in \mathscr{X}\downarrow$, the process $(1, f)$ is an isomorphism if and only if the square $1; f = 1; f$ is an $\mathscr{X}\downarrow$-pullback.*

Proof. $(1, f); (f, 1) = 1$ if and only if there exist x and y in $\mathscr{X}$ such that



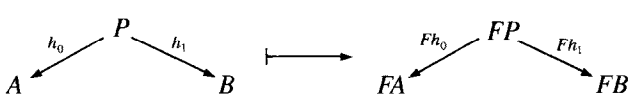
which means that $1; f = 1; f$ is an $\mathscr{X}\downarrow$- pullback. □

It is easy to see that if $\mathscr{X}$ is left-factor closed then any morphism $x \in \mathscr{X}$ has $1; x = 1; x$ an $\mathscr{X}$-pullback. So, for example, any process in $SProc(X)$ with both legs belonging to $\exists^{\not\emptyset}$ is an isomorphism.

3.2. The 2-functor Proc

Here we demonstrate that the process construction is a 2-functor, the domain of which consists of categories with cover systems and functorial structure which interacts well with those cover systems.

If a functor $F : X \to Y$ between model categories induces a functor $Proc(F) : Proc(X, \mathscr{X}) \to Proc(Y, \mathscr{Y})$



between process categories, we will refer to F as being *cover-stable* and write $F : (X, \mathscr{X}) \to (Y, \mathscr{Y})$.

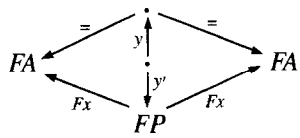
Lemma 17. *The following are equivalent:*

(i) $F:(X,\mathscr{X}) \to (Y,\mathscr{Y})$;

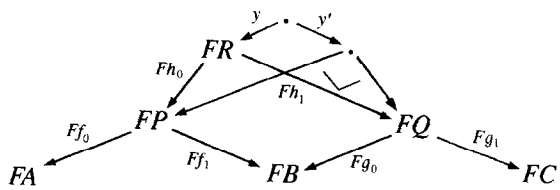
(ii) $F(\mathscr{X}) \subset \mathscr{Y}\downarrow$ *and* F *takes pullbacks to* $\mathscr{Y}\downarrow$*-pullbacks*;

(iii) F *takes* $\mathscr{X}$*-pullbacks to* $\mathscr{Y}\downarrow$*-pullbacks.*

Proof. We first show that (i) and (ii) are equivalent: For any x in $\mathscr{X}$ with codomain A, the span (x,x) is the identity on A. So to preserve equality requires that $\mathscr{X}$ arrows are taken by F to $\mathscr{Y}\downarrow$ arrows:

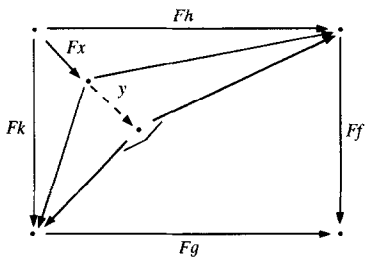


This also suffices to preserve equality as any $\mathscr{Y}\downarrow$-bisimulation determines a $\mathscr{Y}$-bisimulation (Lemma 8). $Proc(F)$ preserves identities as F does, and the composition $(h_0; f_0, h_1; g_1)$ of $f:A \rightsquigarrow B$ and $g:B \rightsquigarrow C$ is preserved exactly when there exist y and y' in $\mathscr{Y}$ such that:



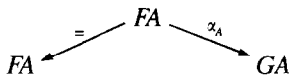
i.e. the induced map to the pullback is in $\mathscr{Y}\downarrow$ and thus $Fh_0; Ff_1 = Fh_1; Fg_0$ is a $\mathscr{Y}\downarrow$-pullback.

To see that (ii) and (iii) are equivalent, suppose F takes $\mathscr{X}$-pullbacks to $\mathscr{Y}\downarrow$-pullbacks. Then the $\mathscr{X}$-pullback $x; 1 = x; 1$ is taken to the $\mathscr{Y}\downarrow$-pullback $Fx; 1 = Fx; 1$, so $Fx \in \mathscr{Y}$. Conversely, if x is witness that $h; f = k;\ g$ is an $\mathscr{X}$-pullback then $Fx; y$ is witness that the outer square below



is a $\mathscr{Y}\downarrow$-pullback. □

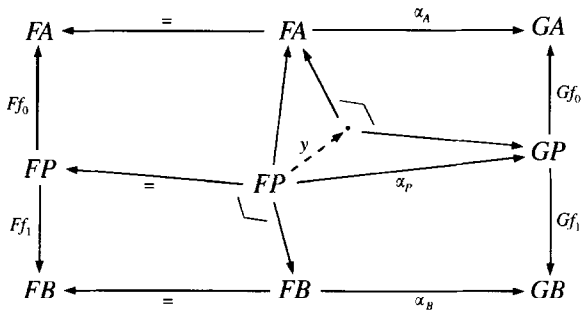
We now consider when a natural transformation $\alpha : F \Rightarrow G : X \to Y$ induces a natural transformation $Proc(\alpha)$



between lifted functors:

Lemma 18. *$Proc(\alpha) : Proc(F) \Rightarrow Proc(G)$ iff the naturality squares of α are $\mathscr{Y}\downarrow$-pullbacks.*

Proof. A naturality square $Proc(F)(f); Proc(\alpha) = Proc(\alpha); Proc(G)(f)$ appears as follows in Y:



To commute in $Proc(Y)$ it is necessary and sufficient that the induced morphism y belongs to $\mathscr{Y}\downarrow$. □

This allows us to define a 2-category $\underline{Cov}$ representing the domain of the process construction.

Definition 19. The 2-category $\underline{Cov}$ is given by

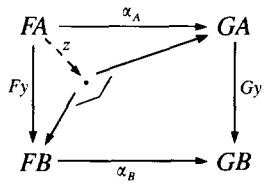
- 0-cells – categories with cover systems $(X, \mathscr{X})$;
- 1-cells – $(X, \mathscr{X}) \to (Y, \mathscr{Y})$ are functors $F : X \to Y$ which are cover-stable in that $\mathscr{X}$-pullbacks are taken to $\mathscr{Y}\downarrow$-pullbacks;
- 2-cells – $F \to G : (X, \mathscr{X}) \to (y, \mathscr{Y})$ are natural transformations which are cover-cartesian in that all naturality squares are $\mathscr{Y}\downarrow$-pullbacks.

As the 1-cells and 2-cells are those of $\underline{Cat}$, to see $\underline{Cov}$ is a 2-category it suffices to show that it is closed to the three forms of composition: cover-stable functors clearly contain identities and are closed to composition; cover-cartesian natural transformations contain natural isomorphism and are closed to composition by Lemma 10; finally, recalling that the horizontal composite $\alpha ;; \beta$ of $\alpha : F \Rightarrow F'$ and $\beta : G \Rightarrow G'$ in $\underline{Cat}$ is given by $\beta_F ; G'\alpha = G\alpha ; \beta_{F'}$, it is clear that $\underline{Cov}$ is closed to this as well.

The following will be convenient for showing that a natural transformation is cover-cartesian.

Lemma 20. *If $\mathscr{X}$ is left-factor closed, then $\alpha : F \Rightarrow G$ is $\mathscr{X}$-cartesian whenever α_A is in $\mathscr{X}$ for all A in Y.*

Proof. Consider the naturality square associated with any $y : A \to B$:



The arrow opposite α_B in the pullback square is in $\mathscr{X}$ and thus the induced arrow z is in $\mathscr{X}$ by left-factor closure. □

Finally, the construction of processes can be viewed as a 2-functor.

Proposition 21. *$Proc : \underline{Cov} \to \underline{Cat}$ is a 2-functor which preserves products.*

Proof. It is clear that *Proc* preserves 1-cell composition (i.e. $Proc(F\,;;\,G)$ is $Proc(F)\,;;\,Proc(G)$) and vertical 2-cell composition (i.e. $Proc(\alpha); Proc(\beta)$ is $Proc(\alpha;\beta)$). To see that horizontal composition of 2-cells is preserved, suppose $\alpha : F \Rightarrow F'$ and $\beta : G \Rightarrow G'$. Then

$$\begin{aligned} Proc(\alpha\,;;\,\beta) &\equiv Proc(\beta; G'\alpha) \\ &= Proc(\beta); Proc(G'\alpha) \\ &= Proc(\beta); Proc(G')(Proc(\alpha)) \\ &\equiv Proc(\alpha)\,;;\,Proc(\beta) \end{aligned}$$

as required.

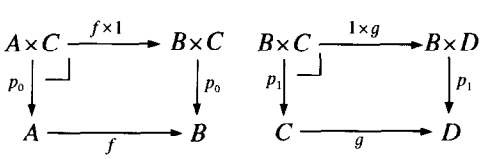
To see preservation of products, note that the bicategories $\overline{Proc}(X \times Y,\ \mathscr{X} \times \mathscr{Y})$ and $\overline{Proc}(X, \mathscr{X}) \times \overline{Proc}(Y, \mathscr{Y})$ are the same. □

3.3. Linear process categories

Clearly the choice of $\underline{Cat}$ as the codomain of *Proc* is not optimal, and begs the question "what structure characterizes a category of processes". Here we show that one can obtain process categories which model various aspects of linear logic by placing certain demands upon the model category and its cover system.

3.3.1. Compact-closure

Products in any category preserve pullbacks and also preserve any cover system since the following squares are always pullbacks:

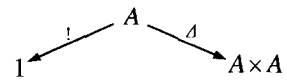


Thus the product functor induces a symmetric tensor on the process category (the associativity, symmetry and unit elimination are cartesian as they are natural isomorphisms). As the diagonal Δ is not cartesian, the lifted tensor is unlikely to be a product. However, it does admit the degenerate form of $\star$-autonomy in which the tensor and cotensor coincide [3].

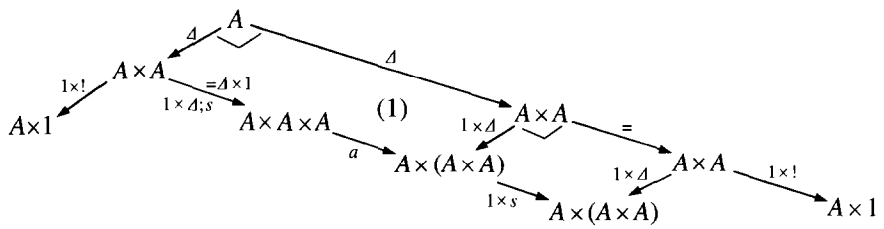
Let X have finite limits and cover system $\mathscr{X}$.

Proposition 22. *$Proc(X,\mathscr{X})$ is compact-closed.*

Proof. Identifying the tensor and cotensor with the lifted product gives a trivial example of a weakly distributive category [9] where the weak distributions are isomorphisms (e.g. $\delta_R^L = s;a;1\otimes s : A\otimes(B\otimes C)\to B\otimes(A\otimes C)$). The involution $-^{\perp}$ is span reversal, and the complementation morphism $\tau_A : \top \to A\otimes A$ is given by the span



The only requirement is $1_{A\otimes\top} = 1\otimes\tau;\delta_R^L;1\otimes\tau^{\perp}$, which is witnessed by the following span in X:



Here we note that (1) is always a pullback and $\Delta;1\times!$ is an isomorphism. □

For example, the categories $Span(X)$ and $Rel(E)$ of spans and relations are compact-closed for finitely complete X and regular E. Furthermore, $SProc(X)$ is compact-closed for any finitely complete X.

3.3.2. *Biproducts*

Coproducts in a lextensive category C, give rise to biproducts in $Span(C)$ [16]. If C is a lextensive category with a cover system $\mathscr{X}$,

Proposition 23. *$Proc(C,\mathscr{X})$ has finite biproducts provided $+$ preserves $\mathscr{X}$.*

Proof. In a lextensive category, $+$ is stable and the injections and codiagonal are cartesian. As $+$ preserves $\mathscr{X}$, it is also $\mathscr{X}$-stable. Thus coproducts lift to $Proc(C,\mathscr{X})$ and, as the process category is self-dual, are simultaneously products. □

Unfortunately, $Tran(C)$ does not inherit coproducts pointwise from C: in a coproduct $A+B$ the initial states of A and B must be coalesced, which would require C to have pushouts along "elements". Even if C had such pushouts, the induced coproduct would not be disjoint (and thus not extensive) – if the initial states of A and B are reachable, $A+B$ allows execution paths which alternate between A and B.

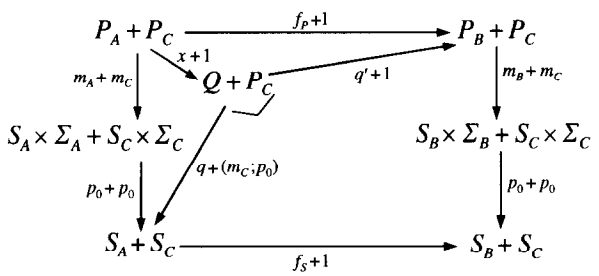
By separating the initial state from the rest of the state space, one obtains a related category of transition systems which is lextensive:

Definition 24. $STran(C)$ is the category of models in C of the following sketch:

These transition systems, in effect, have initial transitions rather than an initial state: the initial transitions of the coproduct $A+B$ are just the sum of the initial transitions of A and B. As a cover system on $STran(C)$ we take the class of morphisms $\exists^{\mathscr{X}}$ which are $\mathscr{X}$-cartesian for both $m^0; p_0$ and $m; p_0$.[5] Note that any f in $\exists^{\mathscr{X}}$ has the component f_{P^0} between the initial transitions belonging to $\mathscr{X}$.

Lemma 25. *Coproducts in $STran(C)$ preserve $\exists^{\mathscr{X}}$ whenever coproducts in C preserve $\mathscr{X}$.*

Proof. We show that $f:A\to B$ in $\exists^{\mathscr{X}}$ implies $f+1:A+C\to B+C$ is in $\exists^{\mathscr{X}}$. Note that $m_{A+C}\equiv m_A+m_C;\langle b_0\times b_0\mid b_1\times b_1\rangle$ and thus $m_{A+C};p_0=m_a+m_C;p_0+p_0$. Letting x be the witness that f is in $\exists^{\mathscr{C}}$ and (Q,q,q') the relevant pullback in C,



we see that $f+1$ is in $\exists^{\mathscr{X}}$ as coproducts in C preserve pullbacks and $\mathscr{X}$. □

Since all functors preserve retractions and isomorphisms, we have finite biproducts in each of the categories $Proc(STran(C),\exists^{\mathscr{R}})$ and $Proc(STran(C),\exists^{\mathscr{I}})$.

[5] We will use the symbol $\exists^{\mathscr{X}}$ to denote the cover system for behavioral equivalence in each of the categories of transition systems, relying on context to disambiguate.

3.3.3. Storage

An inexpensive way of obtaining the exponential type is to use the Fock construction [4]: i.e. in a compact-closed category with finite/countable biproducts $!A \equiv \sum_{n\in\omega} \otimes_s^n A$, where $\otimes_s^n A$ is the nth symmetric tensor power of A obtained by coequilizing the group $\mathscr{S}_n$ of permutations on $\otimes^n A$

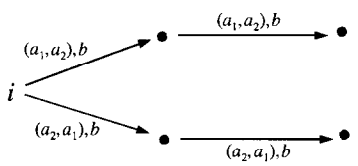
$$\otimes^n A \underset{\sigma_n!}{\overset{\sigma_1}{\rightrightarrows}} \otimes^n A \xrightarrow{\ q\ } \otimes_s^n A$$

In *Rel*, for example, these permutations and their coequilizers are given directly (i.e. as trivial spans) by the corresponding permutations and coequilizers on the product in *Set*:

$$A^n \underset{\sigma_n!}{\overset{\sigma_1}{\rightrightarrows}} A^n \xrightarrow{\ q\ } M_n(A)$$

Given any $r : \otimes^n A \to B$ in *Rel* which satisfies $\forall \sigma \in \mathscr{S}_n.\ \sigma; r = r$, the mediating morphism $\otimes_s^n A \to B$ is given by $q^\circ; r$. Thus $!(-)$ in *Rel* is given by the underlying multiset monad $M(A) \equiv \sum_{n\in\omega} M_n(A)$ on *Set*: the functor $!(-)$ takes a relation (f_0, f_1) to the relation (Mf_0, Mf_1).

Contrary to earlier suggestions [1], the Fock construction fails in *SProc*. To see this, let A and B be the trace specifications generated by the regular expressions $(a_1|a_2)^*$ and b^*, respectively. The process $r : A \otimes A \to B$ given by the transition system

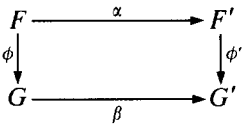


satisfies $c; r = r$. In [10] it is shown that *SProc* is isomorphic to a process category constructed upon *Trace(Set)* – i.e. the category whose objects are those of *SProc* and whose morphisms $A \to B$ are functions $\Sigma_A \to \Sigma_B$ which preserve traces. Forming the coequilizer $q : A \times A \to M_2(A)$ of the identity and symmetry morphisms in *Trace(Set)* gives the obvious candidate for the coequilizer in *SProc*: i.e. the process $1_{A\otimes A}[((a_1,a_2),(a_1,a_2)) \mapsto ((a_1,a_2),\{|a_1,a_2|\})]$, using the notation of [1]. However, $q; q^\circ; r$ is not bisimilar, or trace equivalent, to r as the trace $((a_1,a_2),b)((a_2,a_1),b)$ is admitted only by the former.

The fact that the Fock construction fails does not mean one cannot interpret the exponential type (using multisets). If one has a monad which lies in the domain of the process construction, then the corresponding monad on the process category (which is simultaneously a comonad) provides a potential candidate to model $!(-)$. The following shows how one might obtain such a monad.

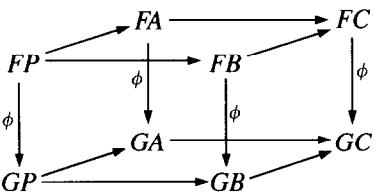
Proposition 26. *Let* $F : (X, \mathscr{X}) \to (Y, \mathscr{Y})$ *and let* $\phi : F \Rightarrow G : X \to Y$ *be pointwise* $\mathscr{Y}\!\downarrow$ *and* $\mathscr{Y}\!\downarrow$*-cartesian. Then*

(i) $G: (X,\mathscr{X}) \to (Y,\mathscr{Y})$ *is* $\mathscr{X}$*-stable*;
(ii) *If* $\alpha: F \Rightarrow F'$ *and* $\phi': F' \Rightarrow G'$ *are* $\mathscr{Y}\downarrow$*-cartesian and*



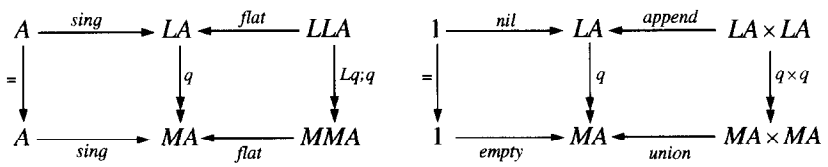
then β *is* $\mathscr{Y}\downarrow$*-cartesian.*

Proof. For (i), note that if the top face of the cube



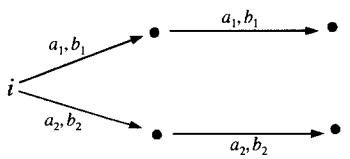
is a $\mathscr{Y}\downarrow$-pullback then the bottom face is also a $\mathscr{Y}\downarrow$-pullback by Lemma 10. Part (ii) follows from the same lemma. □

In *Set*, multisets stand in such a relation to lists. The natural transformation $q: L \Rightarrow M$, given by the sum $\sum_{n\in\omega} q_n$ of coequilizers of $\mathscr{S}_n$, transmits the monad and monoid structure of lists to multisets:



This natural transformation is pointwise $\mathscr{E}$ (regular epimorphism) and is also $\mathscr{E}$-cartesian. Since L is a cartesian monad, M is an $\mathscr{E}$-cartesian monad by the preceeding lemma. Finally, the isomorphisms $MA \times MB \to M(A+B)$ and $1 \to M0$ in *Set* give isomorphisms $!A \otimes !B \to !(A \times B)$ and $I \to !1$ in *Rel*.

Unfortunately, the multiset monad on the category *Trace*(*Set*) is not $\exists^{\mathscr{R}}$-stable since the corresponding $!(-)$ is not a functor in *SProc*. To see this, consider the specifications $A \equiv (a_1|a_2)^*$ and $B \equiv (b_1|b_2)^*$ and the process $f: A \to B$ given by the following transition system:



The process $!f; !f^\circ$ and $!(f; f^\circ)$ are not bisimilar as only the former can deadlock after exhibiting the action $(\{|a_1,a_2|\},\{|a_1,a_2|\})$. Thus $!(-)$ does not preserve composition.

Note that the preceeding example does not exclude the possibility that !(−) is a functor under *trace equivalence*. Indeed, if we quotient *SProc* by trace equivalence rather than bisimulation the resulting process category does have a model of !(−) given by multisets. This fact is most easily seen by taking as a model category the category of *trees* [13].

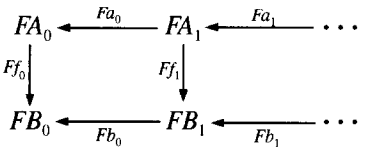
Definition 27. *Tree* (X) is the category of models in X of the following sketch:

$$A_0 \xleftarrow{a_0} A_1 \xleftarrow{a_1} A_2 \xleftarrow{a_2} \cdots$$

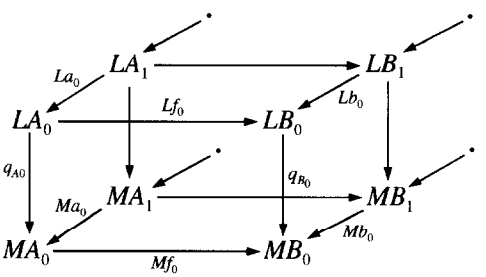
In *Tree*(*Set*), for instance, a tree has an implicit root and corresponds to a trace specification in which every transition is uniquely labeled (and there are no unused labels). The cover system for trace equivalence in this setting is simply the regular epimorphisms. Thus we define the category of synchronous processes modulo trace equivalence as the category of relations upon *Tree*(*Set*):

$$SProc_T \equiv Proc(Tree(Set), \mathscr{E})$$

To see the construction of !(−) in this setting, note that *Tree*(*Set*) is a topos [13] and that lists and multisets are inherited pointwise from *Set*: any functor $F: X \to Y$ gives a functor $Tree(F): Tree(X) \to Tree(Y)$



and for any $\alpha: F \Rightarrow G: X \to Y$ the family $\{(\alpha_{A_0}, \alpha_{A_1}, \ldots) \mid A \in Tree(Y)\}$ is a natural transformation $Tree(F) \Rightarrow Tree(G)$. In *Tree*(*Set*), the inherited $Tree(q): Tree(L) \to Tree(M)$ is in $\mathscr{E}$ and is $\mathscr{E}$-cartesian as this is true pointwise in *Set*:



This makes the monad *Tree*(*M*) and associated monoid structure $\mathscr{E}$-cartesian. Again the isomorphisms $1 \to M0$ and $MA \times MB \to M(A+B)$ given pointwise induce the expected isomorphisms in the process category.

Proposition 28. *$SProc_T$ is a linear category.*

This construction can be performed in a more general setting. For example, the multiset functor exists in any locos in which each object can be totally ordered.

4. Equivalence of process categories

One actually has considerable latitude in choosing a model category upon which to construct a particular processes category. This section demonstrates that the same process category. *SProc*, actually arises from a wide variety of related model categories – including each of the standard models of interleaved concurrency [21]. The proof is based on the following observation:

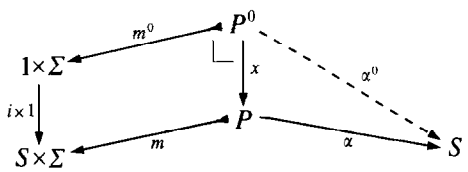
Proposition 29. *If $\mathscr{Y}_L \subseteq \mathscr{Y}$ is left-factor closed and $\alpha: F \Rightarrow G: (X,\mathscr{X}) \to (Y,\mathscr{Y})$ has α_A in $\mathscr{Y}_L$ for all A, then Proc(α) is a natural isomorphism.*

An adjunction between model categories which lies in $\underline{Cov}$ induces an adjoint equivalence between the associated process categories whenever the unit and counit belong to left-factor closed cover systems at each point. In fact, one need not have an adjunction between model categories to obtain an equivalence of process categories: it suffices to have the appropriate functors and natural transformations which belong to left-factor closed cover systems.

4.1. Moving between system models

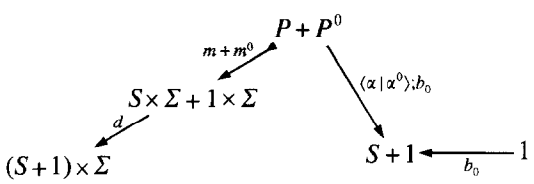
We begin by showing that the transition systems of Section 3.3.2 with separated initial states yield the same process category as ordinary (deterministic) transition systems.

Given an object A of $Tran(C)$ we can extract the initial transitions and thereby form an object GA of $STran(C)$:



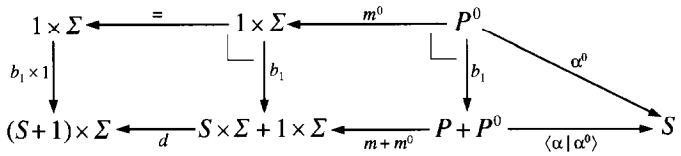
To see G preserves $\exists^{\mathscr{X}}$, recall that the arrows required to be $\mathscr{X}$-cartesian in GA are m; p_0 and m^0; p_0: the former is given and the latter is the pullback of m; p_0 along i.

Given B in $STran(C)$ we obtain an object FB of $Tran(C)$ by adding the initial state to the rest of the state space:

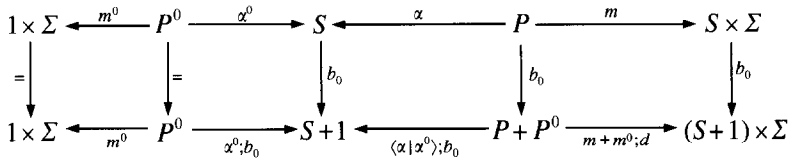


To see F preserves $\exists^{\mathscr{X}}$, note that the arrow required to be $\mathscr{C}$-cartesian in FB is the sum $(m^0; p_0) + (m; p_0)$.

These functors determine an adjunction in $\underline{Cov}$. The separated transition system GFB differs from B in that it contains an unreachable copy of the initial state. Note first that the initial transitions of FB are extracted as follows:

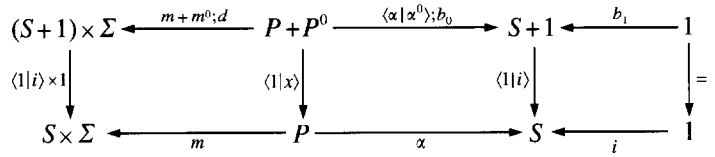


The unit η_B is thus the injection:



This morphism belongs to $\exists^{\mathscr{I}}$ since it is the identity on initial transitions and b_0 is cartesian.

Concerning the counit, the transition system FGA differs from A in that it has a new initial state which is bisimilar to the old (still reachable) initial state. The counit ε_A collapses the new initial state onto the original:



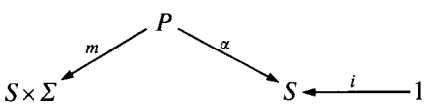
This morphism is in $\exists^{\mathscr{I}}$ also as coproducts are stable and ∇ is cartesian.

As $\exists^{\mathscr{I}}$ is left-factor closed, η and ε are $\exists^{\mathscr{I}}$-cartesian and so this adjunction exists in $\underline{Cov}$. Furthermore, given any lextensive category C and cover system $\mathscr{X}$ preserved by coproducts:

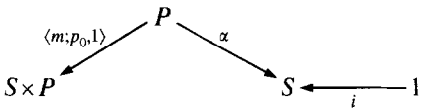
Proposition 30. *$Proc(STran(C), \exists^{\mathscr{X}})$ is equivalent to $Proc(Tran(C), \exists^{\mathscr{X}})$.*

We now turn to nondeterministic transition systems. These can be obtained from deterministic transition systems by simply dropping the condition that the permission set is a subobject (viz. m is not monic):

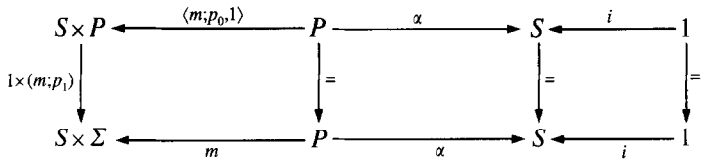
Definition 31. *$NTran(C)$* is the category of models in C of the sketch:



Given a nondeterministic transition system A, one forms a deterministic transition FA system by adding stage information to the labels:



Note that $\langle f, 1\rangle$ is always monic, and F preserves $\exists^{\mathscr{X}}$ as $\langle m; p_0, 1\rangle; p_0 = m; p_0$. This functor is not an adjoint to the inclusion functor $I: Tran(C) \to NTran(C)$, but the morphism



serves both as a natural transformation $IFA \Rightarrow A$ and $FIB \Rightarrow B$. It is pointwise in $\exists^{\mathscr{I}}$ as it has identity effect on states and transitions. Consequently,

Proposition 32. *$Proc(NTran(C), \exists^{\mathscr{X}})$ is equivalent to $Proc(Tran(C), \exists^{\mathscr{X}})$.*

Note that categories of deterministic and nondeterministic transition systems describe in [21] are not the categories discussed above. The former are the Kleisli categories of the *delay* monads on $Tran(Set)$ and $NTran(Set)$, respectively. These monads add a new label, say *, and an *idle* transition on * at every state.

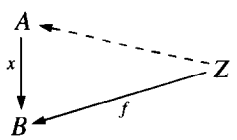
4.2. Behavior models

In [21], it is shown that certain categories *system* models (those which represent process states explicitly) have as coreflective subcategories certain *behavior* models (those which abstract from such details). Here we provide an abstract notion of behavior and show that, for an appropriate cover system, it is sufficient to consider the category of behaviors when constructing a process category. We further link the existence of such subcategories of behaviors to the existence of inductive data types in span categories.

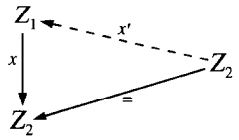
4.2.1. Abstract behaviors

Let $\mathscr{X}$ be a left-factor closed cover system on C:

Definition 33. An object Z of C is an $\mathscr{X}$-behavior if given any $x: A \to B$ in $\mathscr{X}$, every $f: Z \to B$ factors uniquely through x:

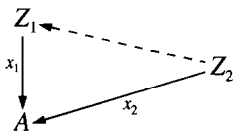


Intuitively, an $\mathscr{X}$-behavior cannot distinguish objects which are $\mathscr{X}$-equivalent. Note that any $\mathscr{X}$-morphism $x: Z_1 \to Z_2$ between $\mathscr{X}$-behaviors is an isomorphism: x has a left inverse x'



which (by left-factor closure) belongs to $\mathscr{X}$. Similarly x' has a left-inverse and is thus an isomorphism.

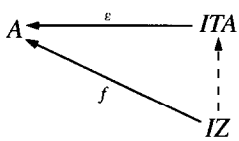
If Z is an $\mathscr{X}$-behavior and $Z \to A$ an $\mathscr{X}$-morphism then we regard Z as an *$\mathscr{X}$-behavior of A*. By left-factor closure and the preceeding observation we see that all $\mathscr{X}$-behaviors for a given object are isomorphic:



A category C is said to have *enough $\mathscr{X}$-behaviors* if for every object A there is an $\mathscr{X}$-behavior TA and a morphism $\varepsilon: TA \to A$ in $\mathscr{X}$.

Proposition 34. *If C has enough $\mathscr{X}$-behaviors then T is a coreflective subcategory with counit ε.*

Proof. The couniversal diagram is an instance of the defining property of $\mathscr{X}$-behavior:



where I is the inclusion functor. □

If C has enough $\mathscr{X}$-behaviors then, for an appropriate extension of $\mathscr{X}$, the process categories constructed upon C and the full coreflective subcategory C^T coincide. We write $\mathscr{Y}^T$ for the restriction of $\mathscr{Y}$ to C^T:

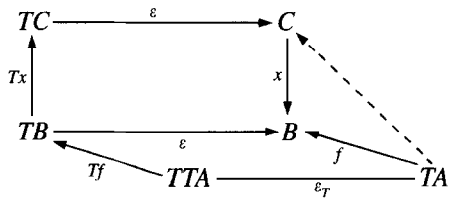
Proposition 35. *If $\mathscr{X} \subseteq \mathscr{Y}$ and T preserves $\mathscr{Y}$ then $Proc(C, \mathscr{Y})$ is equivalent to $Proc(C^T, \mathscr{Y}^T)$.*

Proof. T is $\mathscr{Y}$-stable since it is stable (it is a right adjoint) and preserves $\mathscr{Y}$. As the counit ε is pointwise $\mathscr{X}$ it is (by Lemma 20) $\mathscr{X}$-cartesian and thus $\mathscr{Y}$-cartesian. The coreflection is thus preserved by the process construction and (by Proposition 29) becomes an equivalence. □

Any coreflective subcategory T of C determines a cover system for which T gives enough behaviors: the class $T^{-1}(\mathscr{I})$ (the morphisms of C taken by T to isomorphisms) is a left-factor closed cover system and, as T is an idempotent comonad [5], the counit ε belongs to $T^{-1}(\mathscr{I})$. In fact,

Proposition 36. *If $\mathscr{X} \subseteq T^{-1}(\mathscr{I})$ is left-factor closed and contains ε then:*
(i) *C has enough $\mathscr{X}$-behaviors;*
(ii) $T^{-1}(\mathscr{I}) = \mathscr{X}\downarrow$.

Proof. (i) Let $f: TA \to B$ and let $x: C \to B$ belong to $\mathscr{X}$. Since Tx and ε_T are isomorphisms, there is a morphism $f': TA \to C$ given by $\varepsilon_{TA}^{-1}; Tf; Tx^{-1}; \varepsilon_C$ which (by naturality) satisfies $f'; x = f$.

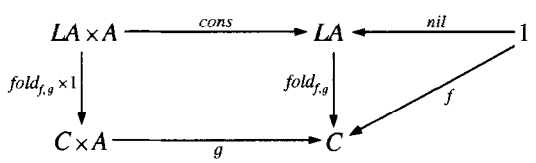


If g also satisfies $f = g; x$ then $g = \varepsilon_{TA}^{-1}; Tg; \varepsilon_C = \varepsilon_{TA}^{-1}; Tf; Tx^{-1}; \varepsilon_C$ as required.

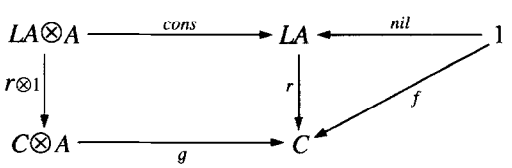
(ii) If f is in $T^{-1}(\mathscr{I})$ then f is in $\mathscr{X}\downarrow$ since $\varepsilon; f = Tf; \varepsilon$ is in $\mathscr{X}$. Conversely, let x be witness that f is in $\mathscr{X}\downarrow$. Then Tx and $Tx; Tf$ are isomorphisms and thus Tf is an isomorphism. □

4.2.2. Behaviors of transition systems

A *locos* C is a lextensive category with list construction (see [7]): i.e. there is an endofunctor L and natural transformations $nil: 1 \to LA$ and $cons: LA \times A \to LA$ with the property that given any $f: 1 \to C$ and $g: C \times A \to C$ there exists a unique morphism $fold_{f,g}$ such that

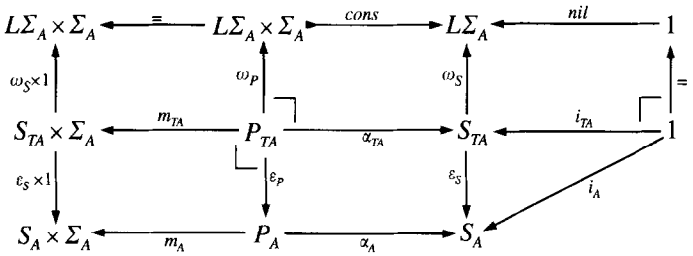


We say that a locos C has *span list construction* if given any $f: 1 \to C$ and $g: C \otimes A \to C$ in $Span(C)$ there exists r such that



commutes in $Span(C)$ and, furthermore, given $q: LA \to C$ and $\alpha: cons;;q \Rightarrow q \otimes 1;;g$ there exists a unique $\beta: q \Rightarrow r$ such that $\alpha;(\beta \otimes 1;;g) = cons;;\beta$.

Let C be a locos with span list construction. Each transition system A in C then induces the following structure in C:



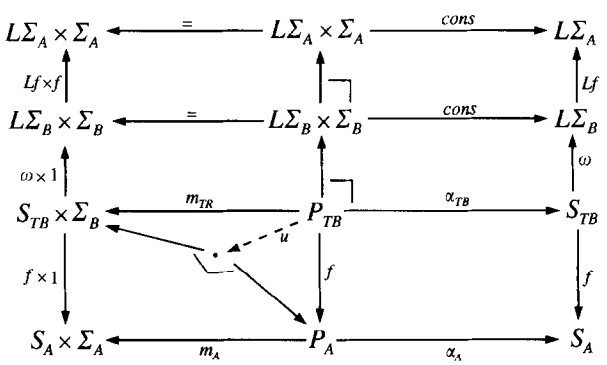
We define the transition system TA – the behavior machine of A – and the morphisms $\varepsilon_A: TA \to A$ and $\omega_A: TA \to LA$ of transition systems as indicated (LA denotes the free transition system on the alphabet of A).

For a deterministic transition system in Set, the behavior machine is the Hoare language [12] generated by A: i.e. S_{TA} is the nonempty and prefix-closed set of strings $\{a_1 \ldots a_N \mid \exists s_1, \ldots, s_n.\ i \xrightarrow{a_1} s_1 \to \cdots \xrightarrow{a_n} s_n\}$, and TA admits a transition $s \xrightarrow{a} sa$ iff $sa \in S_{TA}$. For a nondeterministic transition system, TA is the synchronization tree generated by A: i.e. the corresponding (iso-similar) transition system in which every state is reachable from the initial state by exactly one sequence of $n \geqslant 0$ transitions. It is clear that synchronization trees differ from traces only in that the former do not identify states with strings. Although for different morphisms, it is shown in [21] that the category of traces (resp. synchronization trees) is a coreflective subcategory of deterministic (resp. nondeterministic) transition systems.

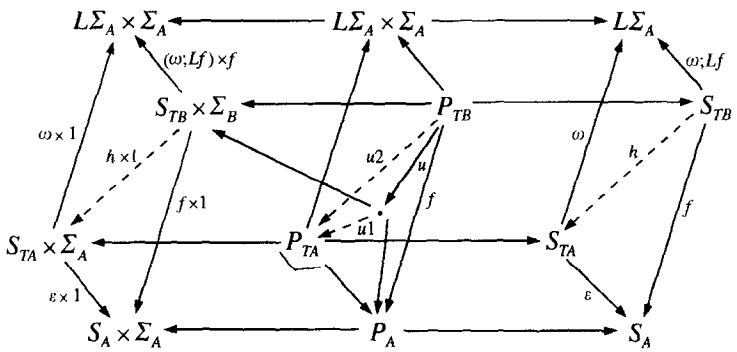
Taking $Trace(C)$ and $STree(C)$ to be the full subcategories of behavior machines in $Tran(C)$ and $NTran(C)$, respectively:

Proposition 37. *$Trace(C)$ (resp. $STree(C)$) is a coreflective subcategory of $Tran(C)$ (resp. $NTran(C)$).*

Proof. For any morphism $f: TB \to A$ of transition systems, one has in C the following structure:

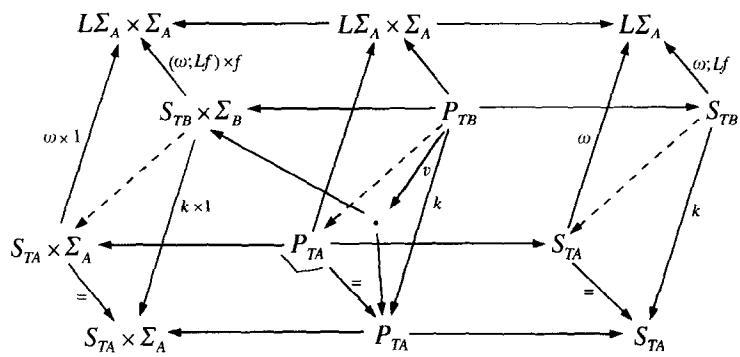


which is a 2-cell $u : cons\,;;\,(\omega_B; Lf, f) \Rightarrow (\omega_B \times 1; Lf \times f, f \times f)\,;;\,(m_A, \alpha_A)$ in $Span(C)$. By the universal property of span list construction, there exists $h : S_{TB} \to S_{TA}$ such that:



where the morphisms labeled u_1 and u_2 are the 2-cells $h \times 1\,;;(m_A, \alpha_A)$ and $cons\,;;h$, respectively. This gives a morphism $h : TB \to TA$ of transition systems satisfying $h; \varepsilon_A = f$.

If k is such that $k; \varepsilon_A = f$ then once more by the universal property of span list construction we have the following commuting diagram in C:

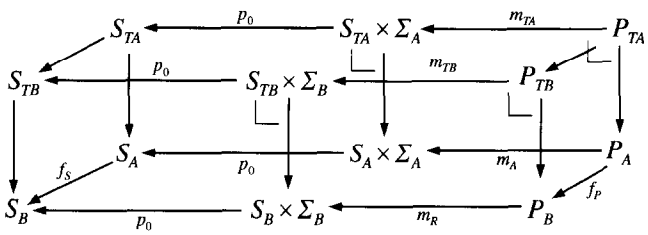


Consequently, k gives a 2-cell satisfying $u; k \times 1\,;;(m_A, \alpha_A) = cons\,;;k$ which forces $k = h$. □

Now, what is the cover system for which traces/synchronization trees are the behaviors of transition systems? First note that the cover systems $\exists^{\mathscr{X}}$ on behaviors are simply restrictions of the corresponding cover systems on transition systems. Then for any cover system $\mathscr{X}$ on C,

Lemma 38. *The behavior functors T preserve $\exists^{\mathscr{X}}$.*

Proof. Any morphism $f : A \to B$ of transition systems gives the following structure in C:

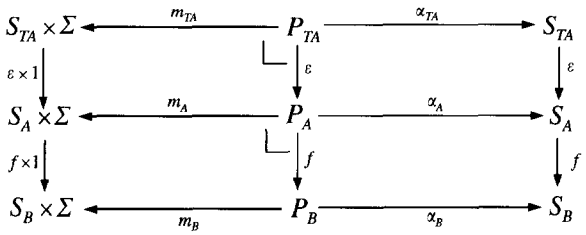


By Lemma 10, if the bottom is an $\mathscr{X}$-pullback then the top will be an $\mathscr{X}$-pullback also. □

The $\exists^{\mathscr{I}}$-morphisms (i.e. the local isomorphisms) of behavior machines are isomorphisms of states, but not necessarily isomorphisms of labels. The (fibration) functor δ which takes a transition system to its labels, is stable and thus allows one to restrict the $\exists^{\mathscr{I}}$-morphisms of transition systems to those whose label components are isomorphisms:

Proposition 39. *Tran(C) and NTran(C) have* $\exists^{\mathscr{I}} \cap \delta^{-1}(\mathscr{I})$*-behaviors given by Trace(C) and Stree(C), respectively.*

Proof. It is enough to see that for $f : A \to B$ in $\exists^{\mathscr{I}} \cap \delta^{-1}(\mathscr{I})$, any behavior machine of A is a behavior machine of B:

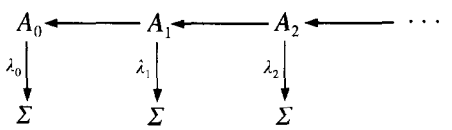


assuming for simplicity that $\Sigma_A = \Sigma_B = \Sigma$. □

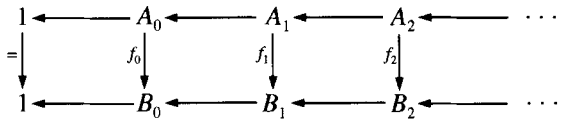
As $\exists^{\mathscr{I}}$ is contained in $\exists^{\mathscr{X}}$ for any $\mathscr{X}$:

Proposition 40. *For the cover systems* $\exists^{\mathscr{X}}$*, the process categories constructed upon each of the following are equivalent: Tran(C), NTran(C), Trace(C) and STree(C).* □

In fact, one can construct an equivalent process category upon an even simpler model category. To see this, note that an equivalent presentation of synchronization trees is as labeled trees [20]: i.e. $STree(X)$ is the category of models in X of the following sketch:

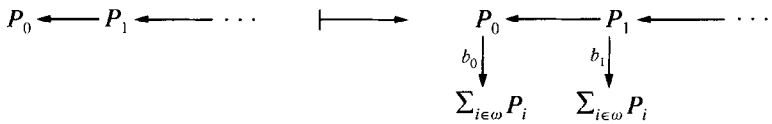


As in the category $Tree(X)$ of trees in X, the cover system $\exists^{\mathscr{X}}$ consists of the morphisms $f : A \to B$ for which all of the squares below are $\mathscr{X}$-pullbacks:

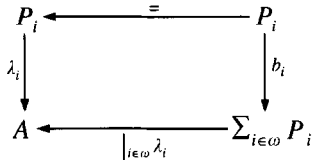


In $Tree(X)$, the $\exists^{\mathscr{X}}$-morphisms are pointwise $\mathscr{X}$. Thus, for example, $\exists^{\mathscr{E}} \subseteq \mathscr{E}$ and $\exists^{\mathscr{I}} \subseteq \mathscr{I}$.

$Tree(C)$ is a coreflective subcategory of both $STree(C)$ and $Trace(X)$): the functor U which forgets the labeling is right adjoint to the (inclusion) functor



which views each "action" as distinctly labeled. The counit is the identity on "states" and on labels is the copairing of the labelings of the original synchronization tree:



Considering the composite coreflections between transition systems and trees, the functors preserve $\exists^{\mathscr{I}}$ and the counits are in $\exists^{\mathscr{I}}$. Thus:

Proposition 41. *$Tree(C)$ gives $\exists^{\mathscr{I}}$-behaviors for $Tran(C)$ and $NTran(C)$.*

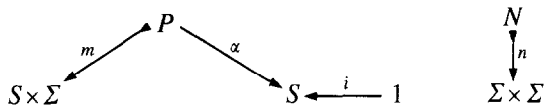
The results of this section show that the only feature of a model category significant for the construction of *SProc* is "extension in time". The standard distinctions "linear vs. branching" and "system vs. behavior" between model categories do not persist to the level of process categories – even explicit labeling of actions is unnecessary.

5. Noninterleaving models and processes

We have shown how all standard models of interleaved concurrency result in the same process category: *SProc*. Here we consider the processes constructed upon a model category for noninterleaved concurrency: the result is a process category which is not *SProc*.

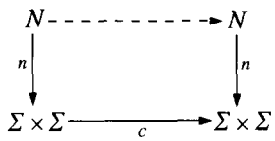
Consider the following variant of deterministic transition systems with independence [21] in which the independence relation is specified on labels rather than transitions. These can be constructed in any category X with finite limits:

Definition 42. *ITran(X)* is the category of models in X of the sketch

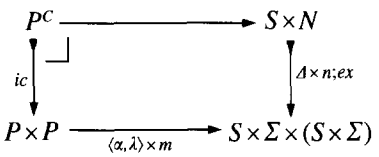


with the following constraints:

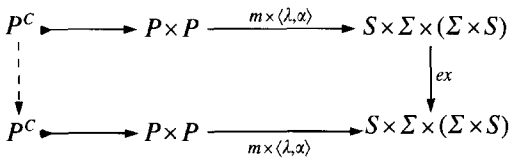
- the independence relation is symmetric,



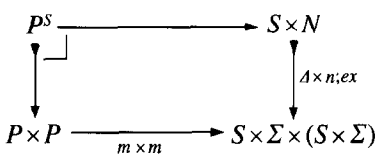
- the independent consecutive transitions



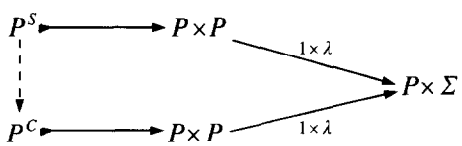
are permitted coherently in either order,



- and the independent sibling transitions



are permitted consecutively



Here we have written λ to represent $m: p_1$.

The requirement in [21] that the independence relation be irreflexive amounts to the statement that there are no idle actions (other than the implicit $*$). Furthermore,

given that morphisms preserve independence, requiring irreflexivity would preclude the existence of a final object.

Proposition 43. *ITran(X) has finite limits.*

As a cover system, one can follow [14] in taking the cover system for bisimulation restricted to those morphisms which reflect consecutive independence. Specifically, define $\exists_I^{\mathscr{R}}$ in to consist of the morphisms of $ITran(C)$ which are $\mathscr{R}$-cartesian for $m: p_0$ and ic.[6]

As in Section 3.3.2, one can modify the sketch of transition systems with independence to obtain a lextensive model category which yields an equivalent process category.

Proposition 44. *$Proc(ITran(C), \exists_I^{\mathscr{R}})$ is compact-closed with finite biproducts.*

Although this yields a process category other than *SProc*, the independence information is of questionable value: as span legs do not reflect independence, actions specified as independent in the interface need not be implemented as such. This can be remedied to an extent by restricting $ITran(C)$ to the subcategory $ITran_R(C)$ whose morphisms reflect consecutive independence. The cover system then becomes simply the one for bisimulation, and we define

$$NSProc(C) \equiv Proc(ITran_R(C), \exists^{\mathscr{R}})$$

Note that $ITran_R(C)$ does not have a final object, and thus $NSProc(C)$ is not compact-closed.

6. Conclusion

We have presented a construction of process categories as categories of generalized relations. The construction begins with a category of process models and a cover system expressing process equivalence. One then forms the category of spans quotiented by the cover system. We have shown how Abramsky's category *SProc* [1] of synchronous processes arises in this way from all of the standard models of interleaved concurrency. The construction is, however, quite general: in [11], it is used to obtain a category of asynchronous processes modulo weak bisimulation.

Previous work [10] had shown that *SProc* is isomorphic to a process category built upon the category of traces. The results of this paper allow a more generous characterization:

[6] Any morphism which is $\mathscr{R}$-cartesian for ic is cartesian for ic, as monics are left-factor closed and monic retractions are isomorphisms.

Theorem 45. *With respect to the cover systems $\exists^{\mathscr{E}}$ for bisimulation equivalence the process categories built upon each of the following model categories are equivalent to SProc:*

- *deterministic transition systems Tran(Set);*
- *prefix-closed languages Trace(Set);*
- *nondeterministic transition systems NTran(Set);*
- *synchronization trees STree(Set);*
- *trees Tree(Set).*

This suggests that *SProc* is a rather robust notion. It also shows that the distinctions "linear vs. branching" and "system vs. behavior" at the level of models disappear at the level of processes – even explicit labeling of actions is unnecessary.

With an appropriate choice of functors and natural transformations, the process construction is 2-functorial. This provides a means of establishing the structure of process categories from the structure of the much simpler model categories. In particular, we show how one may deduce the presence of linear structure in a process category.

In retrospect, the original formulation of *SProc* [1] exacted too much intuition from the category *Rel* of sets and relations. Contrary to the suggestion therein that the Fock construction [4] can be used to model the linear exponential type in *SProc*, we have shown that the required colimits do not exist for either bisimulation or trace equivalence. However, for trace equivalence the multiset functor does extend to the process category and does provide a model of the exponential type.

These results suggest a different spin be applied to the analogy between *SProc* and "relations in time". The category *Tree(Set)* is the category of "sets in time" and (while *SProc* is a process category thereon) the category $SProc_T$ of synchronous processes modulo trace equivalence is its category of relations. $SProc_T$ is thus a more worthy candidate for the title "relations in time".

References

[1] S. Abramsky, S.J. Gay and R. Nagarajan, Interaction categories and the foundation of typed concurrent programming, in: M. Broy, ed., *Deductive Program Design: Proc. 1994 Marktoberdorf Summer School*, NATO ASI Series F: Computer and System Sciences (Springer, Berlin, 1994).

[2] P. Aczel, Non-Well-Founded Sets, CLSI Lecture Notes, Vol. 14 (1988).

[3] M. Barr, **-Autonomous Categories*, Lecture Notes in Mathematics, Vol. 752 (Springer, Berlin, 1979).

[4] M. Barr, *-Autonomous categories and linear logic, *Math. Struct. Comput. Sci.* **1** (1991) 159–178.

[5] M. Barr and C. Wells, *Toposes, Triples and Theories, Grundlehren der mathematischen Wissenschaften*, Vol. 278 (Springer, Berlin, 1985).

[6] J. Bénabou, Introduction to Bicategories, Lecture Notes in Mathematics, Vol. 40 (Springer, Berlin, 1967) 1–77.

[7] J.R.B. Cockett, List-arithmetic open categories: Locoi, *J. Pure Appl. Algebra* **66** (1990) 1–29.

[8] J.R.B. Cockett, Introduction to distributive categories, *Math. Struct. Comput. Sci.* **3** (1993) 277–307.

[9] J.R.B. Cockett and R.A.G. Seely, Weakly Distributive Categories, London Mathematical Society Lecture Notes Series, Vol. 177 (Cambridge University Press, Cambridge, 1992) 45–65.

[10] J.R.B. Cockett and D. Spooner, Sproc categorically, in: *Proc. Concur '94*, Lecture Notes in Computer Science, Vol. 836 (1994) 146–159.

[11] J.R.B. Cockett and D. Spooner, Categories for synchrony and asynchrony, *Electronic Notes Theoret. Comput. Sci.* **1** (1995) 495–520.
[12] C.A.R. Hoare, *Communicating Sequential Processes* (Prentice-Hall, Englewood Cliffs, NJ, 1985).
[13] A. Joyal and I. Moerdijk, *Algebraic Set Theory*, London Mathematical Society Lecture Note Series, Vol. 220 (Cambridge Univ. Press, Cambridge, 1995).
[14] A. Joyal, M. Nielsen and G. Winskel, Bisimulation and open maps, in: *Logic in Computer Science* (IEEE, New York, 1993).
[15] G.M. Kelly, Elementary observations on 2-categorical limits, *Bull. Austral. Math. Soc.* **39** (1989) 301–317.
[16] H. Lindner, A remark on mackey-functors, *Manuscripta Math.* **18** (1976) 273–278.
[17] S. Mac Lane, *Categories for the Working Mathematician* (Springer, Berlin, 1971).
[18] R. Milner, *Communication and Concurrency* (Prentice-Hall, Englewood Cliffs, NJ, 1989).
[19] R. Paré, Simply connected limits, *Canad. J. Math.* **XLII** (4) (1990) 731–746.
[20] D. Pavlović, Convenient categories of processes and simulations 1: modulo strong bisimilarity, Available by ftp from beauty.doc.ic.ac.uk as papers/Pavlovic/CCPS1.ps.gz, January 1995.
[21] V. Sassone, M. Nielsen and G. Winskel, A classification of models for concurrency, in: *Proc. Concur '93*, Lecture Notes in Computer Science, Vol. 715 (Springer, Berlin, 1993) 82–96.
[22] C. Wells, Sketches: Outline with references, Available by ftp from ftp.cwru.edu as math/wells/sketch.ps, February 1994.